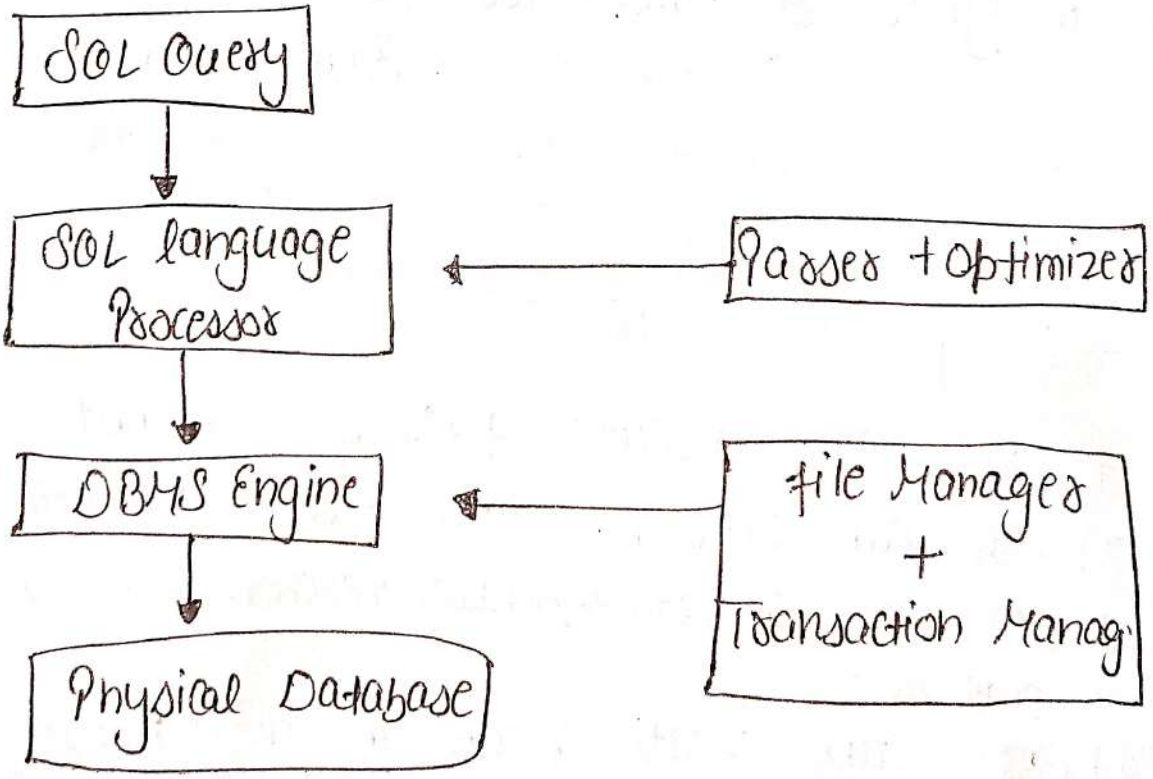
https://python-world-in.stores.instamojo.com/

SQL Process

- When we executing the command of SQL on any relational Database Management System, then the system automatically finds the best routine to carry out our request and the SQL engine.
- Structured Query language contain the following four component.
  1) Query Dispatcher
  2) Optimization Engines
  3) Classic query engines
  4) SQL Query Engine etc.
- All the non-SQL queries are handled by the classic query engine, but SQL query engine won't handle logical files



insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Rules for SQL

- Structure Query Language is not case sensitive. Generally, keywords of SQL are written in uppercase.
- Statements of SQL are dependent on text lines. We can use a single SQL statement on one or multiple text lines
- SQL depends on tuple relational Calculus and relational algebra

Advantages of SQL

1) High Speed
2) No coding needed
3) Well defined Standards
4) Portability
5) Multiple data view

Disadvantage

1) Compler interface
2) Cost
3) Partial Control

Application of SQL

1) Data retrieval and analysis
2) Data Manipulation
3) Database Administration
4) Business inteligence
5) Web development

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

# SQL	No-SQL
1) It is a relational database management system	1) It is a non-relational or distributed database system
2) SQL uses a relational data model with tables	2) Non-SQL databases are uses data Models with various formats.
3) The Schema of SQL databases is pre-defined, fixed, and Static.	3) The Schema of No-SQL databases is a dynamic schema for unstructured data.
4) These databases are vertically scalable	4) These databases are horizontally scalable

SQL Constraints

SQL Constraints are rules that are defined for a table column to ensure the data in the table is accurate, consistent and valid.

- Constraints also help to maintain the integrity of the database
- They applied to a single column or a group of columns in a table.
- There are several types of Constraints in SQL:

1) NOT NULL Constraint :- It ensures that a Column cannot have a NULL value.

insta - @python_world_in

https://python-world-in.stores.jo

2) Unique Constraint: Ensure that each value in a column is unique across all rows in the table.

3) Primary key Constraint: A combination of NOT NULL and UNIQUE constraints that identifies a unique row in a table

4) Foreign key: Ensures that a value in a column matches the primary key of another table

5) Check Constraint: Ensures that the values in a column meet a specific condition.

6) DEFAULT Constraint: Specifies a default value for a column when no value is specified

- Every Constraint start or written as large letter.
- By using these constraints, you can create a structured and reliable database that can prevent data inconsistencies and improve the quality of data

---

Data integrity

Data integrity in RDBMS refers to the accuracy, completeness, and consistency of data stored in a database. It ensures that the data is correct and reliable, and that it can be trusted for use in business operations and decision making.

insta - @python_world_in

https://python-world.in.stores.instamojo.com/

There are several types of data integrity

1) Entity integrity: This ensures that each row in a table is unique and can be identified by a primary key.
   - It prevents duplicate data from being inserted into the table.
2) Referential integrity: Rows cannot be deleted, which are used by other records.
3) Domain integrity: This ensures that data values in each column of a table are within the defined range of valid values or data types.
4) User - defined integrity: This allow the database designes to specify additional constraints or rules to ensure that the data is consistent and accurate.

Data Types

SQL Datatype is used to define the value that a column can contain

Data types are used to represent the nature of the data that can be stored in the database table.

- for ex: in a particular column of a table, if we want to store a String types of data then we will have to declare a String data type of this column.

- String Data types
  Numeric Data types
  Data and time data types.

types of Data types.

- Binary Data type -> [binary, varbinary, image]
- Approximate [float, real], Exact [int, smallint, bit, decimal,

insta - @python_world.in

https://python-world-in.stores.instamojo.com/

→ String Data types - allows to store a group of characters (numbers, letters, symbol, special characters, etc.)

1) Char (Size) → It is used to specify a fixed length string that can contain numbers, letters and special characters.
- Its size can be 0 to 255 characters. Default is 1.

2) VARCHAR (Size) → It is used to specify a variable length string that can contain numbers, letters, and special characters.
- Its size can be from 0 to 65535 characters.

3) BINARY (Size) → It is equal to CHAR() but stores binary byte strings.
- Its size parameter specifies the column length in the bytes. Default is 1.

4) VARBINARY (Size) → It is equal to VARCHAR() but stores binary byte string
- Its size parameter specifies the max. column length in bytes.

5) TEXT (Size) → It holds a string that can conta a maximum length of 255 charac

6) TINY (TEXT) → It holds a string with a maximum length of 255 charact

7) MEDIUMTEXT

8) LONGTEXT

9) SET (val1, val2, val3, ...)

10) BLOB (size)

insta - @python_world_in

https://python-world-in-tamoio.com/

→ Numeric Data types → Used for numeric value - widely used DT in SQL

1) BIT (Size): → It is used for a bit-value type. The number of bits per value is specified in size.
   - It size can be 1 to 64.
2) INT (Size): → It is used for the integer values.
3) FLOAT (Size,d) → It is used to specify a floating point number. Its sizes parameter specifies the total number of digits.
   - The no of digits after the decimal point is specified by d parameter.
4) BOOL → It is used to specify Boolean values true and false.
   - Zero considered false.
   - Non-zero considered True.

[bit, tinyint, smallint, bigint

→ Date and Time Data Types → used for contain both Date and Time

Date: YYY-MM-DD hh:mm:ss

1) DATE: → It is used to specify data format YYYY-MM-DD.
   - It supported range is from '1000-01-01' to '9999-12-31'
2) DATETIME: → It is used to specify date and time combination.
   - '1000-01-01 00:00:00' to '9999-12-31 23:59:59
3) TIMESTAMP (fsp): → It is used to specify the timestamp.
   - Its value is stored as the no of seconds.
   - YYXY-MM-DD hh:mm:ss

insta- @python_world_in

https://python-world.in stores.instamojo.com/

4) TIME(fsp) :→ It is used to specify the time format
- hh:mm:ss

5) Year :→ It is used to specify a year in four-digit format.

Others data types

SQL Server data types:

1) char(n) → fixed width character string data type.
- Size upto 8000 characters

2) varchar(n) → It is a variable width character string data type.

3) varchar(max) :→ variable width character string data types.

4) Text → Variable with character string DT.

5) binary(n) → fixed width binary string data type

SQL	SQL Plus
- A language - ANSI Standard - keyword cannot be abbreviated. - Statement manipulate data and table definations in the database	- An environment - Oracle proprietary - keywords can be abbreviated. - Commands do not allow manipulation of values in the database.

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

#SQL Syntax

- When you want to do some operations on the data in the database, then you must have to write the query in the pre-defined syntax of SQL.
- The syntax of the Structured query language is a unique set of rules and guidelines, which is non case-sensitive.
- Its syntax is defined and maintained by the ISO and ANSI standards.
- Some points about SQL keywords.

1) You can write the keywords of SQL in both uppercase and lowercase, but writing the SQL keywords in uppercase improves the readability of the SQL query.

keywords (eg SELECT, GRANT, DELETE)

[~~select *~~] [SELECT * FROM EMPLOYEES;]

[Select * FROM EMPLOYEES;]

[select * FROM EMPLOYEES;]

above three statement are equal.

- SQL syntax depends on relational algebra and tuple relational calculus.
- SQL statement or syntax are dependent on text lines. We can place a single SQL statement on one or multiple text lines.

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

SQL Statement

SQL Statement tell the database what operation you want to perform on the structured data

[SELECT "Column_name" FROM "table_Name";]

- each SQL Statement begins with any of the SQL keywords and ends with the semicolon (;).
- Used for separating the Multiple SQL Statement which are going to execute in the smal call.

1) SELECT: Retrieves data from one or more tables
2) INSERT INTO: add new records to a table.
3) UPDATE: Modifies existing records in a table.
4) DELETE: removes records from a table.
5) CREATE Table: creates a new table.
6) ALTER TABLE: Modifies the structure of an existing table.
7) ORDER BY: Sorts the data in ascending or descending order based on one or more columns.
8) GROUP BY: groups data based on one or more columns and performs aggregate function like SUM, AVG, COUNT etc.
9) JOIN: Combines data from multiple tables based on a common column or condition
10) DROP TABLE: delete a table and its data from the database.

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

SQL Constraints

SQL constraints are used to specify rules for the data in a table.

- Constraints can be column level or table level
  - → Column level :→ constraints apply to column.
  - → Table level :→ constraints apply to whole table.
- Constraints can be specified:
  1) When a table is created with the CREATE TABLE statement.
  2) We can use a ALTER TABLE statement to create constraints even after the table is created. [If any more constraint added then use ALTER Table]

Syntax:→

```
CREATE TABLE table_name(
    column1 datatype constraints,
    column2 datatype constraints,
    column3 datatype
    .....
);
```

Constraints_name :→ NOT NULL, UNIQUE, PRIMARY KEY.

1) NOT NULL: NULL means empty, i.e, the value is not available.
   - Whenever a table's column is declared as NOT NULL, then the value for that column cannot be empty for any of the table's records.

[NULL does not mean zero. NULL means empty column, not even zero.]

Syntax: Create table table_name (Column_name_1 datatype_1 NOT NULL, Column_name_2 data type_2);

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Example: CREATE TABLE Student
(
ID int (6) NOT NULL,
NAME varchar (10) NOT NULL,
ADDRESS varchar (20)
);

2) UNIQUE: Duplicate value are not allowed in the columns to which the UNIQUE constraint is applied.
- The column with the unique constraint will always contain a unique value.
- This constraint can be applied to one or more than one column of a table, which means more than one unique constraint can exist on a single table.
- Using the UNIQUE constraint, you can also modify the already created tables.

Syntax: Create table table-name (column-name1 data type-1 unique,
column-name-2 data type 2);

Example: CREATE TABLE Student
(
ID int(6) NOT NULL UNIQUE,
NAME varchar (10),
ADDRESS varchar (20),
~~PRIMARY~~ KEY(ID)
);

3) PRIMARY KEY: → Primary key Constraint is a Combination of NOT NULL and Unique constraints.

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

- NOT NULL constraint and a UNIQUE constraint together forms a PRIMARY constraint.
- The column to which we have applied the primary constraint will always contain a unique value and will not allow null values

Syntax: Create table table-name (column-name-1 datatype primary key, column-name-2);

Example: CREATE TABLE Student
(
ID int(6) NOT NULL UNIQUE,
NAME varchar(10),
ADDRESS varchar(20),
PRIMARY KEY (ID)
);

4) FOREIGN KEY: A foreign key is used for referential integrity.
- When we have two tables, and one table take reference from another table, i.e the same column is present in both the tables and that column act as a primary key in one table.

Example: CREATE TABLE Customerss (
ID INT NOT NULL,
Name varchar(20) NOT NULL,
Age int NOT NULL,
Address Char(25),
PRIMARY KEY (ID)
);

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

```
CREATE TABLE Orders
(
ID INT NOT NULL,
DATE DATETIME,
CUSTOMER_ID INT references CUSTOMERS(ID),
AMOUNT double,
PRIMARY KEY(ID)
);
```

5) CHECK: Whenever a check constraint is applied to the table's column, and the user want to insert the value in it, then the value will first be checked for certain condn before inserting the value into that column.

for ex: Create table table_name (column_name, datatype-
CHECK (condition) .....);

Example:
```
CREATE TABLE Student
(
ID int (6) NOT NULL,
NAME varchar (10) NOT NULL,
AGE int NOT NULL CHECK (AGE >=18)
);
```

- The above query creates a table Student and specifies the condn for the fild AGE as (AGE >=18). That is, the user will not be allowed to enter any record in the table with AGE <18.

6) DEFAULT: Whenever a default constraint is applied to the table's column, and the user has not specified the value to be inserted in it, then

insta @python_world_in

https://python-world-interviews.instamojo.com/

default value which was specified while applying the default constraint will be inserted into that particular column.

Syntax: Create table table_name (column_name1 datatype1, column_name2 datatype2 DEFAULT value);

Example:
```
CREATE TABLE Student
(
ID int(6) NOT NULL,
NAME varchar(10) NOT NULL,
AGE int DEFAULT 18
);
```

Example:
```
CREATE TABLE Emp
(
  Emp ID int NOT NULL,
  Name varchar(20) NOT NULL,
  Age int NOT NULL,
  Address varchar(100) DEFAULT
  Salary numeric(10,2) NOT NULL,
PRIMARY KEY (Emp_ID),
FOREGN KEY (Dept_ID), REFERENCES Dept(ID),
CHECK (Age >=18)
);
```

insta - @python_world_in

SQL Syntax

https://python-world-in.stores.instamojo.com/

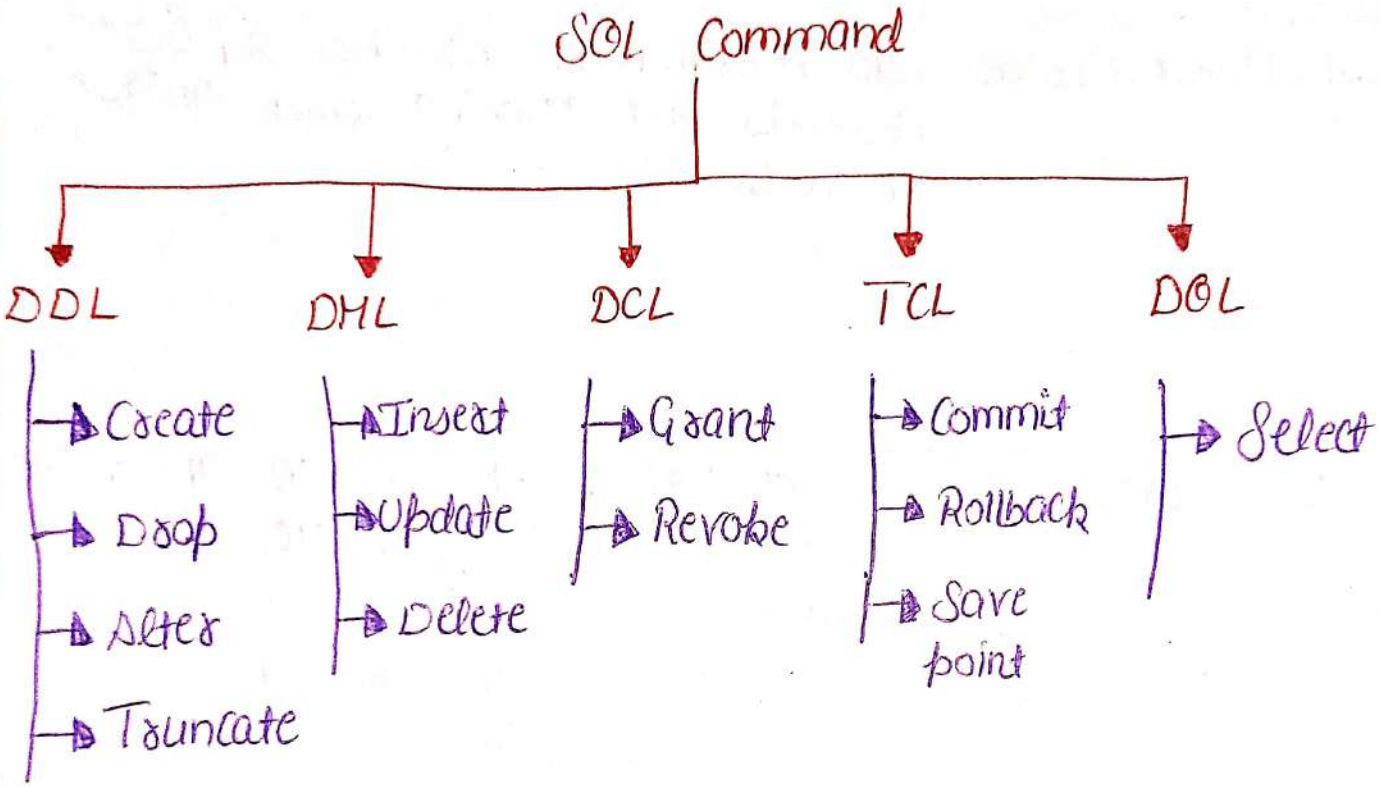
Commands	Syntax
• SELECT Statement:	Select Statement used to retrieves information from a table Syntax: SELECT column1, column2 . . . . column FROM table_name;
• DISTINCT Clause	To Eliminate all the duplicate records and fetching only unique records. Syntax: SELECT DISTINCT column1, col2 . . . FROM table_name;
• IN Clause	Select values the match any valu in a given list of value. Syntax: SELECT Column1, Column2, . . . N FROM table_name; Where Column_name IN (val-1, val-2
• Between Clause	define a range of values tha the column values must fall in to make the cond^n true. Syntax: SELECT Column1, Column2 . . . N FROM table_name WHERE Column_name BETWEEN val AND val-2;
• Like Clause	%, _ Operator SELECT Column1, Column2 . . . N FROM table_name WHERE Column_name LIKE {PATTERN

Insta - @pythonworld_in

https://pythonworldstores.instamojo.com/

SQL Commands

- SQL Commands are instructions. It is used to Communicate with the database.
- It is also used to perform specific tasks, functions and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop, Modify.



1) DDL :- Data Definition language

- It is used for Creating and modifying the database objects such as tables, indics, views.
- DDL Comands are used to define the structure and schema of the database.
- All the command of DDL are auto-committed that means it permanently save all the changes in the database.

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

· Some commands that come under DDL:
  - CREATE
  - ALTER
  - DROP
  - TRUNCATE

---

1) CREATE: to create new table or database. When a table is created its column are name, datatype and sizes are supplied for each column, each table must have atleast one column.

· CREATE Statement can be used for
  → Creating a database
  → Creating a table

→ Creating a table

It specifies column names of the table, it data types (eg: varchar, integer, date etc) and can also specify integrity constraints (eg: primary key, foreign key, unique)

Syntax:

```
CREATE TABLE table-name
(
    column-name1 datatype,
    column-name2 datatype,
    - - - - - - - - -
    column-namen datatype.
    (integrity constraints)
);
```

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Example:

```
CREATE TABLE Emp (
    Emp-ID int,
    Name varchar(20),
    Age int,
    Address varchar(100),
    Salary numeric(10,2)
);
```

Table-name
attributes
datatype
Size
[Length is not fixed]

Output:

Emp-ID	Name	Age	Address	Salary

#Note: · DESC or DESRIBE Statement:

To describe the details of the table structure.

Syntax:- DESC table-name;

Example: DESC Emp; [Create Database database-name;

· SHOW Statement

To see existing databases and tables.

- SHOW database;
- SHOW tables;

· USE Statement

To use or select any existing database.

Syntax: USE database-name;

Example: USE my-db;

insta - @python_world_in

https://python-world.instamojo.com/

2) ALTER: To alter (modify) the structure of table.
- It is also used to add and drop various constraints on an existing table.

---

1) ALTER TABLE - ADD COLUMN
- for adding new column in a table

Syntax: ALTER TABLE table-name
ADD column-name(s) datatype(s); indicate add multiple data and columns

Example: ALTER TABLE Emp
ADD Email varchar(100);

Emp-id	Name	Age	address	Salary	Email

Add new column email

---

2) ALTER TABLE - DROP COLUMN
for removing existing columns in a table

Syntax: ALTER TABLE table_name
DROP column_name(s);

Example: ALTER TABLE Emp
DROP Email;

Emp-id	Name	Age	address	Salary

∴ remove the above column
i.e E-mail.

insta - @python_world_in

https://python-world-in-notes.blogspot.com/

3) ALTER TABLE - MODIFY COLUMN

To modify existing column in a table.
→ To change data type of any column or to modify its size.

Syntax: → ALTER TABLE table_name
MODIFY column_name datatype;

Example: ALTER TABLE Emp
MODIFY Name varchar(100);

It size is [20] change into [100] i.e Modify

Check it with desc command.

---

4) ALTER TABLE - RENAME COLUMN

To rename an existing column in a table

Syntax: → ALTER TABLE table_name
RENAME old_column_name To new column;

Example: ALTER TABLE Emp.
MODIFY Address To Location;

Empid	Name	Age	Location	Salary

[only Column Name Change]

Address rename to location.

insta - @python_world_in

https://pythonworld.in/storage/projects/

3) DROP: It completely removes a table from the database.

- This command will destroy the table structure and the data stored in it. [delete table with structure]

Syntax: DROP TABLE table_name;

Example: DROP TABLE Emp;

- DROP command is also be used as Database.
- To delete the complete database

Syntax: DROP DATABASE database_name;

---

4) TRUNCATE: It used to remove all rows from a table. it is similar to the DELETE statement with no WHERE clause.

[only delete the value not table structure]

Syntax: TRUNCATE TABLE table_name;

Example: TRUNCATE TABLE Emp;

Emp:

Emp_id	Name	Age	Location
1	A	32	Delhi
2	B	25	Pune

↓ Truncate command

Emp id	Name	Age	Location

→ Structure present
→ only records delete

insta - @python_world_in

https://python-world-in-telugu.blogspot.com/

2) DML :- Data manipulation language

- DML commands are used for accessing and manipulating the data stored in the database.
- The DML commands are not auto-committed that means it can't permanently save all the changes in the database.
- They can be rollback

---

1) INSERT :- The INSERT INTO Statement is used to insert new records in a table.

Syntax 1: Specify both the column values and the values to be inserted.

```
INSERT INTO table-name (column1, 2, 3 ... N)
VALUES (value 1, value 2, ---value N)
```

Syntax 2: Specify only values to be inserted. But needed to remember the column order.

```
INSERT INTO table-name
VALUES (value 1, value 2, ---values N);
```

- Uses to Modification of database.

Example: Emp

Emp-ID	Name	Age	Address	Salary
1	Rahul	32	Delhi	20000
2	Karan	25	Pune	30000
3	Kiran	23	Patna	30000
4	Mahi	26	Goa	45000

insta - @python_world_in

https://python_world.instamojo.com/

1st Syntax specifies column

```
INSERT INTO Emp (Emp-ID, Name, Age, Address, Salary)
    VALUES (1, 'Rahul', 32, 'Delhi', 2000);
```

2nd Syntax without column

```
INSERT INTO Emp VALUES (4, 'Chirag', 25, 'Mumbai', 650)
```

To Check

```
SELECT *
FROM Emp;
```
[To see table]

---

2) UPDATE: → To modify the existing record in a table.

- always use the WHERE clause with the UPDATE statement to update the selected rows, otherwise all the rows would be affected.

Syntax

```
UPDATE table_name
SET column1 = value1, column2 = value2 ....
WHERE condition;
```

Example: To update the address of Emp-ID: 3 to Chennai

```
UPDATE Emp
SET address = 'Chennai'
WHERE Emp-ID = 3;
```

Then Move on Emp table.

3	Karan	23	Chennai	2000

```
SELECT *
FROM Emp;
```

↓ Update.

insta - @python_world_in

https://python-world-instagram.instamojo.com/

3) DELETE :- The DELETE statement is used to delete existing records in a table.

Syntax:

```
DELETE FROM table-name
WHERE Condition;
```

- Same WHERE Cause use.

Example: To delete the employee records of Pune location.

```
DELETE FROM Emp
WHERE Address = 'Pune';
```

To Check

```
SELECT *
FROM Emp;
```

Emp-ID	Name	age	Address	Salary
1	Rahul	32	Delhi	20000
3	Kiran	23	Patna	30000
4	Mani	26	Goa	45000

[CRUD] → Create, Read, Update, Delete]

---

3) DCL :- Data Control language

- DCL command are used for controlling the user access in a database
- It is used to control privilege in Databases. i.e it provide right to execute a particular type of SQL.
  - ii) also gives right to connect the database and create a table in your Schema

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

- If you not have privilege not create table.
- Component of SQL.

DCL Commands

1) GRANT :→ Both are used in DDL and DML.
- Used to add permissions to users

Syntax:

```
GRANT <privilege list> | ALL
ON <object>
To <user-list> | Public
[with GRANT OPTION];
```

Example:

```
GRANT ALL ON
Employee To ABC;
[With GRANT OPTION];
```

Where.

Privilege list :→ Specifies the permission to be granted to users like. ALTER, DELETE etc.

- ALL :→ Specifies all the permission.
- object :→ Specifies name of table or their column.
- User-list :→ Specifies name of user Permission gra
- PUBLIC :→ Specifies all users.

With GRANT OPTION :→ Specifies that the user in user.

insta - @python_world_in

https://python-world-in-stores.instamojo.com/

2) REVOKE :- Used to remove permissions from users.

Syntax: REVOKE <privilege-list> | ALL
ON <table name> [<column-comma-list>]
FROM <user-list> | Public;

eg: REVOKE update on employee from user(ABC);

---

4) TCL :- Transaction Control language.

- TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.
- These operations are automatically committed in the database
- The change in the database that we made using DML commands are either performed or rollbacked using TCL.

---

1) COMMIT :- Used to save the change permanently

Syntax: COMMIT;

2) ROLLBACK :- Used to undo one or more changes

Syntax: Rollback;

3) Savepoint :- Used to save the changes temporarily so that it can be rolled back to a particular point

Syntax: SAVEPOINT <savepoint_name>;

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

5) DQL :- Data Query language.

- used to fetch the data from the data-bas

1) SELECT :- used to retrieve data from a database table.

- also called DQL because it is used to query information from a database table.
- It used with different Conditions and CLAUSES

Syntax To retrieve selected field from the table

```
SELECT column1, column2, ... N
FROM table_name(s);
```

Example:

Emp-ID	Name	age	address	Salary
1	A	32	Delhi	20000
2	B	25	Pune	15000
3	C	23	Pune	2000
4	D	25	Mumbai	6500

- SELECT Emp-ID, Name.
  FROM Emp;

Emp-ID	Name.
1	A
2	B
3	C
4	D

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Syntax: To retrieve all the field from table.

Select * → Means all value shows
from Emp;

[SELECT *
FROM table (s);]

↓

Example: [left or complete table form]

# Difference	DDL	DML
full form	Data Definition language	Data manipulation language.
Basic	used to create the database schema.	used to manipulate database
Commands	CREATE, ALTER, DROP etc.	Select, INSERT, UPDATE etc.

# Difference Basis	ALTER	UPDATE
	ALTER command is a DDL command.	UPDATE command is a DML command.
Purpose	ALTER add, delete, modify the attributes of relations in the ~~tab~~ database	UPDATE command modify one or more records in the relations.
values	initializes values.	Specified value
Syntax	—	—

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

DELETE and Drop

Difference	DELETE	DROP
Basic	DELETE remove some or all the tuple from a table.	can remove entire schema from the database.
Language	DML Command	DDL Command.
Clause	WHERE Clause used	No Clause used.
Roll-back.	Can be roll-backed action perform	Can-not be roll-backed action perform.

Delete and Truncate

Diff	DELETE	TRUNCATE
Language	DML	DDL
Basic	Specify the tuple that you want to delete.	deletes all the tuples from a relation.
Speed	Slower	faster.
Restore	followed commit or rollback.	Can't be restore.
Syntax	—	—
	—	—

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Basic Query Structure

- SQL is based on set and relational operation with certain modifications.
- A typical SQL query has the form:

```
Select A1, A2, ... , An
from r1, r2, ... , rn
where P
```

- As represents attributes
- rs represents relations i.e table.
- P is a predicate (or cond^n).

- This SQL query is equivalent to the relational algebra expression.

$$\Pi_{A1, A2 \ldots An}(\sigma_P(r_1 \times r_2 \times \ldots \times r_m)).$$

Clauses

- Clauses are in-built functions available to us in SQL. With the help of clauses, we can deal with data easily stored in the table.
- Clauses help us filter and analyze data quickly. When we have large amount of data stored in the database., we use clauses to query and get data required by the user.
- Some of the examples of Clauses are:→ Where, and, or, like etc.

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

1) GROUP BY:

GROUP BY Statement is used for organizing similar data into groups.

- The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN) to group the result-set by one or more column.

→ GROUP By clause is used with SELECT statement in SQL query

→ GROUP BY clause is place after WHERE Clause SQL

→ GROUP BY clause is placed before ORDER BY Clause in SQL

Syntax:

```
SELECT column_name(s), aggregate-function(column_name)
FROM table-name
[WHERE Condition]
GROUP BY column_name(s)
[ORDER BY column-name(s)];
```

(same. / same.)

Example:

Emp-ID	Name	Age	Dept-ID
1	A	32	500
2	B	25	500
3	C	23	501
4	D	25	502
5	E	24	500
6	F	29	502
7	G	30	501
8	H	31	502

insta -@python_world_in

https://python-world-in.stores.instamojo.com/

· (Group By with COUNT aggregate function)

-> SELECT Dept-id, COUNT(*)
FROM Emp
GROUP BY Dept-id

↓

Dept-id	COUNT(*)
500	3
501	2
502	3

· GROUP By with COUNT and ORDER By)

→ SELECT Dept-id, COUNT(*)
FROM Emp
GROUP BY Dept-id
ORDER BY COUNT(*);

↓

Dept-id	COUNT(*)
501	2
502	3
500	3

---

2) HAVING :-) HAVING Clause is used to specify a search condⁿ for a group or an aggregate.

insta - @python_world_in

https://python-world-in-stores.instamojo.com/

[∴ HAVING Clause apply on groups
∴ WHERE Clause apply on complete table.]

→ HAVING Clause is placed after the GROUP BY Clause.

→ The WHERE Clause place condⁿ on the Selected column, whereas the HAVING Clause places condⁿ on groups created by the GROUP BY Clause

[Make groups, apply condⁿ, which group satisfy the condⁿ, that group result show]

Syntax:

```
SELECT column name(s), aggregate-function()
FROM table-name
[WHERE condition]
GROUP BY column-name(s)
HAVING condition
[ORDER BY column-name(s)];
```

Order of Clause

1) SELECT
2) FROM
3) WHERE [apply on complete table]
4) GROUP BY → apply condⁿ
5) HAVING
6) ORDER BY

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Example:

```
SELECT Dept-id , COUNT(*)
FROM Emp
GROUP BY Dept-id
HAVING COUNT(*) > 2;
```

[Not apply where because we need only groupwise not in complete table]

↓

Dept ID	COUNT(*)
500	3
502	3

---

3) ORDER BY:

The output of the SELECT queries do not have any specific order. The ORDER BY clause allows us to specify order for the query output.

- ORDER BY Clause is used with SELET Statement for arranging retrieved data in sorted order
- The ORDER BY clauses by Default sort the retrieved data in ascending order [ASC]
- The sort data in descending order, [DESC] keywood is used with ORDER BY Clause.

Syntax:

```
SELECT Column-list
FROM table_name
[WHERE Conditions]
ORDER BY Column1, Column2.. ColumnN
[ASC/DESC];
```

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Example Table = Emp

Emp-ID	Name	age	Salary
1	A	32	2000
2	B	25	1500
3	C	23	6500
4	D	26	8500
5	E	22	4500

- Eg: Ascending order - Default [by Salary]

```
SELECT *
FROM Emp
ORDER BY Salary;
```

← add Multiple column [Name] ↓ Primary ordering

[1st Colum Primary ordering and 2nd Colum Secondary ordering

Secondary ordering

Emp-ID	Name	age	Salary
1	B	25	1500
2	A	32	2000
5	E	22	4500
3	C	23	6500
4	D	26	8500

- Descending order by name

```
SELECT *
FROM Emp
ORDER BY Name DESC;
```

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

4) WHERE CLAUSE :- is used to select a particular record based on a condition. It is used to filter records.

- WHERE Clause is used to specify a condition while fetching the data from a single table or by joining with multiple conditions.
- The WHERE Clause is not only used by SELECT statement, but it is also used in the UPDATE, DELETE statement.

Syntax: "SELECT with WHERE Clause"

```
SELECT column1, column2, ... columnN
FROM table-name(s)
WHERE _condition;
```

Example: To retrieve emp-id, Name from Emp tab whose salary is greater than 2000.

```
SELECT Emp-ID, Name
FROM Emp
WHERE Salary > 2000
```

Emp-ID	Name
5	E
3	C
4	D

insta - @python_world_in

https://python-world.in.stores.instamojo.com/

5) DISTINCT Clause :

The SELECT DISTINCT statement is used to return only distinct (different) values. or simply it is used to return only unique values.

- DISTINCT Clause is used to remove duplicates from result set of a SELECT statement.

Syntax: SELECT DISTINCT column 1, column 2 --- N
FROM table-name;

Ex: SELECT without DISTINCT

Previous Table use.

```
SELECT Address
FROM Emp;
```

Address
Delhi
Pune
Pune
Mumbai
Patna

- SELECT with DISTINCT

```
SELECT DISTINCT Address
FROM Emp;
```

Address
Delhi
Pune
Mumbai
Patna

→ only unique values show.

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

AND Operator

- WHERE Clause can be combined AND operator.
- It is used to combine multiple cond^n with WHERE clause, in SELECT, UPDATE, DELETE statements.
- All the cond^n must be satisfied for a record to be selected. [True]

Syntax:
```
SELECT column1, column2, ... columnN
FROM table_name
WHERE Condition1 AND Condition 2 ... AND cond^nN;
```
Ex:- Salary > 200 AND Age < 25;

OR operator

- C1 or C2 are True cond^n.
- Any one of the cond^n must be satisfied with record to be selected.

Syntax:
```
SELECT column1, column 2, .. columnN
FROM table_name
WHERE Condition1 OR Condition 2 .. ;
```
eg:- Salary > 2000 OR Age < 25;

NOT operator

- It is used to negate the cond^n with the WHERE Clause, in SELECT, UPDATE or DELETE statement.
- It displays a record if the cond^n(s) is NOT TRUE. [NOT IN, NOT between]

Syntax:
```
SELECT column1, column 2, .. columnN
FROM table_name
WHERE NOT Condition;
```

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

LIKE operator

- The LIKE operator is used to WHERE Clause to search for a specified pattern in a column.
- The LIKE operator is used to match string pattern values using two wildcard characters
- Wildcard Characters : It is used to substitute one or more Characters in a String
  - The percent sign (%) :- It represent zero, one or multiple characters.
  - The underscore (_) :- It represent only a single character.

Syntax

```
SELECT column1, column2, ....
FROM table-name.
WHERE columnN LIKE pattern;
```

Ex: It Select all Employee with a Name Starting with "K"

```
SELECT * FROM Emp.
WHERE Name Like "K%";
```

→ with a Name that have 'a" in 2nd posi.

```
SELECT * FROM Emp.
WHERE Name LIKE "_a%";
```

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

SQL operators

The SQL reserved words and characters are called operators, which are used with a WHERE clause in a SQL query.

- Operators are used to specify cond^{n} in an SQL statement.

1) Arithmetic operators
2) Comparison operators
3) Logical operators.

1) Arithmetic operators :→ It perform the mathematical operation on the numerical data of the SQL tables.

1) SQL Addition operator (+)
2) SQL Subtraction operator (-)
3) SQL Multiplication operator (*)
4) SQL Division operator (/)
5) SQL Modulus operator (%)

1) SQL Addition operator (+): It is used to perform addition operation on the data items, items include either single column or Multiple columns.

Syntax: operand 1 [+] operand 2;

Example: SELECT emp-id, emp-name, Salary, Salary+100 AS "Salary +100" FROM addition;

(Table Name: addition; Column name: Salary+100)

insta - @python_world_in

https://python-world-in-stores.instamojo.com/

Output:

emp id	emp-name	salary	salary +100
1	A	25000	25100
2	B	25000	25100
3	C	52000	52100

→ operation applied on single column.

2) Subtraction operator:- Subtract right hand operand from left hand operand.

· Implementation

```
SELECT emp-id, emp-name, salary, salary -100
    AS "Salary -100" FROM SubStraction;
```

Example same not salary minus.

3) Multiplication operator:- It is use to perform multiplication of data.

```
# SELECT emp-id, emp-name, salary, salary *100
    AS "Salary *100" FROM addition;
```

5) Divion operator:- Divides left hand operand by right hand operand

Same implementation "Salary /100"

6) Modulus operator:- It is use to get remainder when one data is divided by another.

Salary % 25000 ← Take one data from table

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

2) Comparison operators

The comparison operators in SQL compare two different data of SQL table and check whether they are the same, greater and lesser. The SQL comparison operators are used with the WHERE clause in the SQL queries.

1) (=) Equal
2) (!=) Not equal
3) (>) Greater than
4) (>=) Greater than or equal to
5) (<=) less than or equal to
6) (<) Less than.

1) Equal operator :→ (=).
- This operator is highly used in SQL queries.
- Check if the values of two operands are equal or not, if yes then condn becomes true.

Syntax: SELECT * FROM TABLE_NAME WHERE ATTRIBUTE Condition_Operator Given_Value

Ex:→

Emp-Id	Emp Name	Emp Salary
201	A	3000
202	B	4000
203	C	3000
204	D	3000

insta - @python_world_in

https://python-world.instamojo.com/

SELECT * FROM Salary WHERE Emp-Salary = 3000

Output:

Emp-id	Emp-Name	Emp-Salary
201	A	3000

2) (!=):
↳ Check if the value of two operands are equal or not, if values are not equal then cond^n becomes True.

Syntax: SELECT * FROM Emp-details WHERE Salary != 40;

---

3) (>):
↳ Greather than, Check if the value of two operand is greates than the value of right operand, if yes then cond^n becomes true

Syntax: SELECT * FROM Emp-details WHERE Salary >30

output:

Emp-id	Emp-Name	Emp-Salary
202	B	4000

4) (<):
↳ Less than, Check if the value of two operand is greates than the value of right operand, if yes then cond^n became true.

Query:→ SELECT * FROM Emp-details WHERE Salary < 40K.

output: less than 40 Salary Employee Show

insta - @python_world_in

https://python-world-in-tamil.instamojo.com/

5) >= :) check if the value of left operand is greater than or equal to the value of right operand, if yes then condn becomes true

Query: SELECT * FROM Emp-details WHERE Salary >= 40k;

Ex: also include 40 k value or other greather value

6) <= :) Same defination.

- SELECT * FROM Emp-details WHERE Salary <= 40k;

Ex: Same above concept

---

3) Logical operator :) The logical operators are those that are true or false. They return true or false values to combine one or more true or false values.

1) ALL [compare a value to all value in another value set]
2) AND [Both T and F Boolen Table]
3) OR " " "
4) NOT [only one Boolean expression Table]
5) ANY
6) BETWEEN
7) IN
8) EXISTS
9) SOME
10) UNIQUE [No duplicacy]

insta - @python_world_in

https://python-world.instamojo.com/

String functions

SQL String functions are the performed pre-defined functions that allows the database users for string manipulation.

- String functions are used to perform an operation on input string and return an output string.

1) ASCII ():→ Return the ASCII value for the specific characters.

- Syntax: ASCII ('A');

Output → A → 65

2) Char-length () → Returns the length of a string (in characters).

- Syntax: Select Char-length ("Tannu");

output = 5 [also count space]

3) Concat():→ Add two or more expression together

- Syntax:→ Select Concat ('Tannu', 'Dhanda');

- Output :→ TannuDhanda

1) Concat_ws():→ Add two or more expression together with a separator

Syntax: Concat_ws ('_', 'Tannu', 'Dhanda');

Output: Tannu_Dhanda.

;) field() , find-in-set() , format() , Insert()

insta @python_world_in

https://python-world-in.stores.instamojo.com/

Set operators

The SQL Set operation is used to combine the two or more SQL SELECT Statements.

1) Union
2) Intersect
3) Minus

1) Union :→ The SQL Union operation is used to combine the result of two or more SQL SELECT Queries
- It eliminate the duplicate row.

Syntax:
```
SELECT column-name FROM table1
UNION
SELECT Column-name FROM table2;
```

Ex :→

Id	Name
1	A
2	B
3	C

1st table.

ID	Name
3	C
4	D
5	E

2nd Table.

```
SELECT * FROM first
UNION
SELECT * FROM second;
```

ID	Name
1	A
2	B
3	C
4	D
5	E

insta - @python_world_in

https://python_world_in.stores.instamojo.com/

2) Intersect :- It is used to combine two SELECT statement. The intersect operation returns the common rows from both the SELECT statements.

- Syntax:
```
SELECT column-name FROM table1
INTERSECT
SELECT column-name FROM table2;
```
- Show unique value.

3) Minus :- It combines the result of two SELECT statement. Minus operator is used to display the rows which are present in the first query but absent in the second query.

- Syntax:
```
SELECT Column - name FROM table1
MINUS
SELECT Column - name FROM table2;
```

Ex:

Id	name
1	A
2	B

Aggregate functions

Aggregate functions performs calculations on set of values and return a single values.

- Aggregate functions ignore NULL values, expect Count(*).

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

1) COUNT()
2) SUM()
3) AVG()
4) MAX()
5) MIN()

- Aggregate functions are often used by GROUP BY clause of the SELECT statement to group multiple rows together as input to form a single value output

---

1) COUNT() function: ⇒ Count function returns the number of rows that matches the specified criterian.

- Syntax: SELECT COUNT(column name) [* → all data]
  FROM table-name
  WHERE condition;

- eg: ⇒ Total no of student records

  SELECT COUNT(*)
  FROM Student;

COUNT(*)
6

ID	Name	Masks
1	A	90
2	B	40
3	C	70
4	D	60
5	E	70
6	f	NULL

- ignore NULL value
- count of entered masks.

  SELECT COUNT(Masks)
  FROM Student; →

COUNT(Masks)
5

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

2) SUM():→ SUM function returns total sum of a selected numeric columns.

Syntax:
```
SUM (column-name)
FROM table-name
WHERE condition;
```

Ex: sum of student marks

```
SELECT SUM(Marks)
FROM student;
```

→

SUM(Marks)
330

- If apply (Distinct) → Duplicate value don't take.

---

3) AVG():→ Average function returns the average value of selected numeric column.

Syntax:
```
SELECT AVG (column-name)
FROM table-name
WHERE Condition;
```

Ex:→ average mark of student

```
SELECT AVG(Marks)
FROM students;
```

→

AVG(Marks)
66

AVG(Marks) = SUM/Count(Marks)

$= \frac{330}{5} = 66.$

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

4) MAX() :- Maximum function returns maximum values of selected column.

Syntax:
```
SELECT MAX (Column-name)
FROM table-name
WHERE Condition;
```

Example:
```
SELECT MAX (masks)
FROM Student;
```

MAX(Masks)
90

---

5) MIN() :- Minimum function returns minimum value of selected column.

Syntax:
```
SELECT MIN (Column-name)
FROM table-name
WHERE Condition;
```

Example:
```
SELECT MIN (Masks)
FROM Student;
```

---

6) SQRT() :- It is used to find out the squar root of any number.

Syntax:
```
SELECT SQRT (NUMBER);
```

Example:
```
SELECT SQRT(25);
```

insta - @python_world_in

https://pythonworld-in.stores.instamojo.com/

Indexes

- The index in SQL is a special table used to speed up the searching of the data in the database tables.
- The index concept in SQL is same as the index concept in the noral of a book.
- It is the best SQL technique for improving the performance of queries.
- The drawback of using indexes

They slow down the execution time of UPDATE and INSERT statements.

- Advantage:-

they speed up the execution time of SELECT and WHERE statements.

→ How it is created

CREATE INDEX Statement is used to create indexes in tables and to indicate wheather the index is in ascending or descending order

Syntax:

```
CREATE INDEX index_name ON Table_name (column);
```

→ Index_name:→ name of that index that we want to create.

→ Table_name:→ name of the table on which the index is to be created.

→ Column_name:→ represents the name of the column on which index is to be applied.

Insta - @python_world_in

https://python_world_in.storeinstamojo.com/

If we want to create an index on the combination of two or more column, then the syntax:

```
CREATE INDEX Index_name ON Table_Name (column(s));
```

- Create UNIQUE INDEX

Unique index is the same as the primary key in SQL. The unique index does not allow selecting those columns which contain duplicate values.

- It maintain data integrity.

Syntax:

```
CREATE UNIQUE INDEX index name ON Table Name;
```

→ When should indexes be created

- A column contains a wide range of value.
- Does not contain a large numbers of null values.
- One or more columns are frequently used together in a where clause.

- When Should indexes be avoided

- The table is small.
- The column are not often used as a cond^{n} in the query.
- The column is updated frequently.

insta - @python_world_in

https://pythonworld-in.myinstamojo.com/

Types of SQL indexing

1) Clustered index
2) Non-Clustered index

1) Clustered index:- Clustered index is the index that will arrange the rows physically in the memory in sorted order.

- When Primary key is created automatically the Clustered Index is created.
- We can create only one Clustered on a table.

Example:

```
CREATE TABLE Employee-Index
(
Emp-ID INT,
Emp-name VARCHAR(100),
Emp-Sal int
)
```

---- Insert data into table ----

```
INSERT INTO Employee-index VALUES (1,'Tannu',1000)
INSRT INTO  "      "     "     (5,'Tanu2',1050)
INSERT INTO "      "     "     (7,'Tnu',1570)
```

Select * from employee-index

output

Emp id	Name	Sal
1	Tank	1000
3	Tan 2	1050
2	Tn	1570

→ Not create a Primary key Then data is unsorted order

insta - @python_world_in

https://python-world.instamojo.com/

Now create Clustered index

Create CLUSTERED index ix_Employee_index on Employee_index (Emp_ID)

Then, data is come into sorted order

1	Tom	1000
2	T—	1050
3	S—L	1570

← output

2) Non-Clustered : 1) When a Unique key is created automatically, when Non-Clustered index is created.

- A non-Clustered index is an index that will not arrange the rows physically in the memory in sorted order.
- We can create one or more Non-Clustered index in a table.

Ex: Same above example.

Not sorted element

- Then use Un-Clustered syntax

Create NonClustered index ix_table name on table_name (column_name ASC).

- after apply Non-Clustered index order isn't sorted.

DROP INDEX Command

Sy:→ DROP INDEX index_name;

Ex: DROP INDEX abc;

- When performance may be slowed or improved.

Insta - @python_world_in

https://python-world-in-insta.blogspot.com

1) Single-Column indexes:→ create only one table column.

Syntax:→ CREATE INDEX index_name
ON table_name (column_name);

2) Unique indexes:→ Not used for performance, but also used for data integrity.
- Not allow duplicates values.

Syntax: CREATE INDEX index_name
ON table_name (column_name);

3) Composite indexes:→ is an index on two or more columns of a table.

Syntax: CREATE INDEX index_name
ON table_name (column1, column2);

4) Implicit indexes:→ that are automatically created by the database server when an object is created.

Tables in SQL

Table is a collection of data, organized in terms of rows and columns.
- In DBMS, table is known as relation

Syntax: CREATE table "table_name"
(
"column1" "data type",
"column2" "data type",
....
"columnN" "data type");

insta - @python_world_in

https://python_world_in.stores.instamojo.com/

Eg: employee table

Emp-Name	Address	Salary
T	Lucknow	15K
B	Allahbad	18K
C	Delhi	20K

- In the above table, "Employee" is the table name, "EMP_NAME", "ADDRESS" and "SALARY" are the column names.
- The combination of data of multiple columns form a row.
  eg:→ 'A', 'Delhi' and 20K are the data of one row
- The SQL Table variable is used to create, modify, rename, copy and delete tables.

Type of Tables in SQL Server

1) User Tables (Regular Tables)
2) Local Temporary Tables
3) Global Temporary Table.
4) Table variable

Views

Any relation that is made visible to a user as a "virtual relation" is called a View.

- It provides limited access to DB
- Views are the virtual relations, through which a selective portion of data from one or more relation (tables) can be seen.

@python_world_in

https://python_world-in.stores.instamojo.com/

• Uses of View

1) It helps in query processing
2) to hide data complexity.

• View are Stored in the data dictionary in the tables called USER-VIEWS.

→ Define

A view is defined using the Create view Statement.

Create view v as <query expression>

→ In SQL:

View are kind of virtual tables.

• A views also has rows and columns as they are in a real table in the database.

• We can create a view by selecting fields from one or more tables present in the database.

Tables

Student Details

S-ID	Name	Address
1	ABC	Delhi
2	CDE	Noida
3	EFG	Punjab
4	GHI	Bihar

Student Marks

Marks	Age
90	19
90	13
79	19
70	21

Same Column

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Creating Views

- We can create view using CREATE VIEW statement.
- A view can be created from a single table or multiple tables

Syntax

```
CREATE VIEW view-name AS
SELECT column1, column2.....
FROM table_name
WHERE condition;
```

view-name : Name of the view
table-name : Name of table
condition : condn to select rows.

-> View Creating from a Single table

- Create a view named Details view from the table Student Details.

Query:

```
CREATE VIEW Details View AS
SELECT Name, Address,
FROM Student Details
WHERE S-ID < 2;
```

- To see the data in the view, we can query the view in the same manners.

```
SELECT * FROM Detail View;
```

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Output:

Name	Address
ABC	Delhi
CDE	Noida.

-> Creating View from multiple tables

- Two tables [Student Details and Student Marks]

Query -> CREATE VIEW Marks View AS
SELECT StudentDetails.Name, StudentDetails.Add,
StudentMarks
FROM StudentDetails, StudentMarks
WHERE StudentDetails.Name = StudentMarks.Name;

- To display data of View Marks View:
SELECT * FROM Marks View;

Output:

Name	Address	Marks
ABC	Delhi	90
CDE	Noida	80

Deleting Views

In SQL, we can delete or drop a view using the DROP Statement.

Syntax: DROP VIEW view-name;

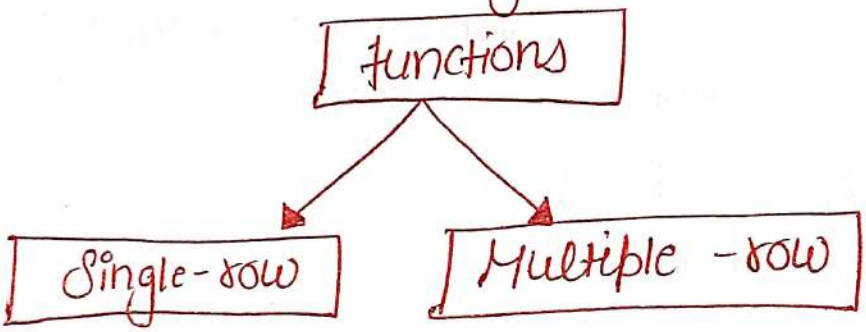
view-name -> which you want to delete.

Insta - @python_world_in

https://pythonworld.in.stores.instamojo.com/

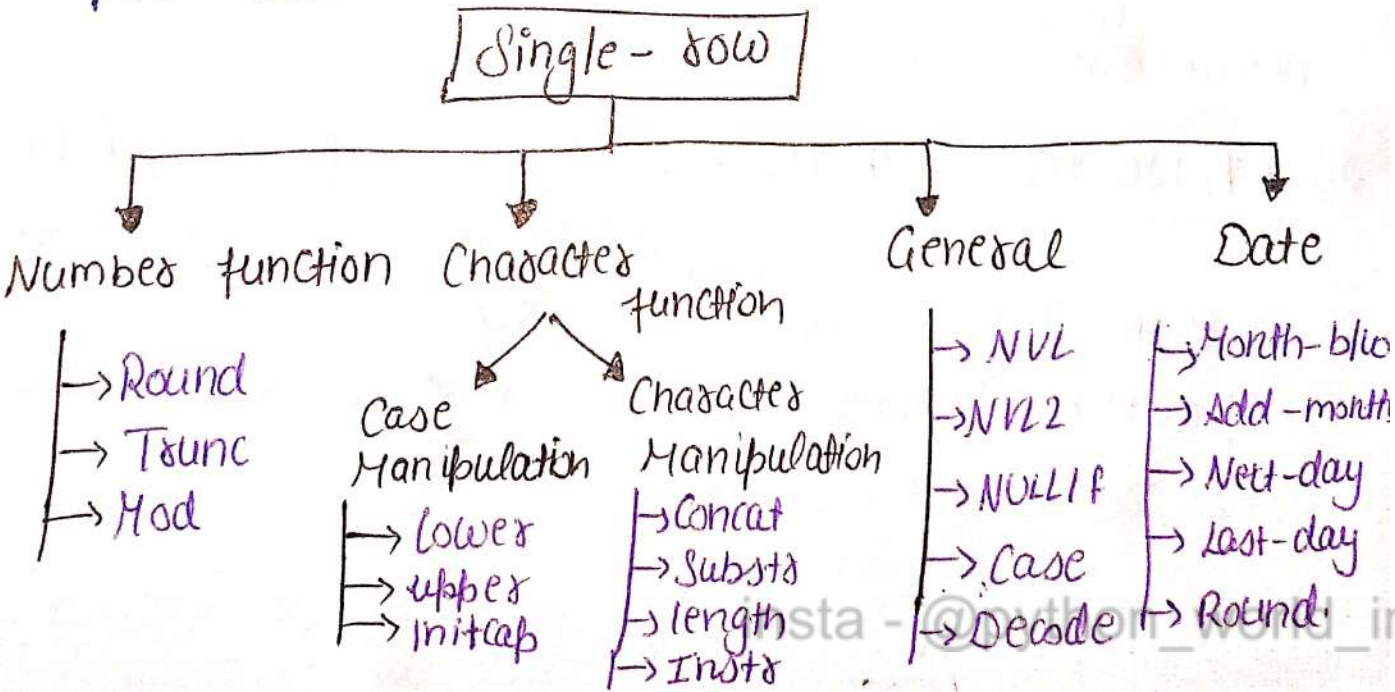
Functions in SQL

- Functions are the object in SQL which take one or more input from user, take operation on input and produce one or more output.
- A function is a special type of predefined command set that performs some operation and returns a single value.
- Function operates on zero, one, two or more values that are provided to them. The values that are provided to functions are called parameters and arguments.



1) Single-row functions :

Single row functions are the one who works on single row and return one output per row.



insta - @python_world_in

Function in SQL

- Row functions can be character function, functions, data functions and conversion functions. [Used to manipulate data items]
- Single row function can be used in SELECT Statement, WHERE and ORDER BY clause.

- Single row functions

1) Number functions: Accepts numeric input and returns numeric values.
   - function under the category are ROUND, TRUNC and MOD
   - ROUND and TRUNC function are used to round and truncate the number value.

   eg:-> SELECT ROUND (1372.427)
   FROM dual;

   ROUND (1372.427)

   1372.5

   - MOD is used to return the remainder of the division operation b/w two numbers.

---

2) Character function: 1) Case Conversion function
   Accepts character input and returns a character value.
   - function under the category are UPPER, LOWER and INITCAP

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

- UPPER function: Converts a string to upper case

eg:→ SQL COURSE → SQL COURSE
[String] [Upper function]

- Lower function: Converts a string to lower case

eg:→ SQL Course → sql course

- INITCAP function: Convert first letter to each word to uppercase and remaining letters to lowercase.

eg:→ SQL Course → Sql Course

2) Character functions: Accept character input and return number or character value.

- functions are CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD.

- CONCAT function: Joins value together

eg:→ ('Good', 'String') → GoodString

- SUBSTR function: return a portion of a string from a given start point to an end point.

- LENGTH function: Show the length of a string as a numeric value

eg:→ ('String') → 6

- INSTR function: finds the numeric position of a named character

eg:→ ('String', 'r') → 3)

insta - @python_world_in

https://python-world-in.stackroute.com/

3) General function :- Usually contains NULL handling functions. The functions under

NVL, NVL2, NULLIF, CASE, DECODE

4) Date function :- Date asithmetic operations return date or numeric values function under the category are MONTHS - B/w, ADD, MONTS, NEXT - DAY, LAST - DAY.

- MONTHS - BETWEEN function returns the count of months b/w the two dates
- ADD - months function add 'n' number of months to an input date
- NEXT - DAY function returns the next day of the date specified

2) Multi - Row functions :- These functions are also called "Aggregate functions" or group functions

- All these row functions will process multiple records.
- Applied on group of records and returns one value from the entire group.

→ Aggregate functions

1) MAX()
2) MIN()
3) AVG()
4) COUNT()

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Nested Queries

- also known as Subquery and inner queries
- The Subquery or inner query is an SQL query placed inside another SQL query. [a query is written inside a query. The result of inner query is used in execution of outer query.
- It is embedded in the Having or WHERE Clause of the SQL Statements.

. Important rule

1) The SQL Subqueries can be used with the following statement along with the SQL expression operators.
   - SELECT Statement [operators like
   - UPDATE Statement =, <, >, >= etc]
   - INSERT Statement
   - DELETE Statement

2) The Subqueries in SQL are always enclosed in the parenthesis and placed on the right side of the operators.

⇒ There are mainly two type of nested queries

1) Independent Nested Queries:] In independent nested queries, the execution order is from the innermost query to the outer query.
   - operators such as IN, NOT IN, ALL are use to write independent nested queries.

   IN → If a column value in the outer query's result is present in the inner query's result

insta - @python_world_in

https://python_world_in_stores.instamojo.com/

- NOT IN: Checks if a column value in the outer query's result is not present in the inner query result

Executed last

```
→ Select id, Name
  FROM Employees
  WHERE ID IN (SELECT Emp-ID FROM AWARDS)
```

Nested Query [Executed first]

2) Co-related Nested Queries: the inner query uses the values from the outer query to execute the inner query for every row processed by the outer query.

Syntax

```
SELECT Column_name
FROM table-name
WHERE Column_name expression operator
(SELECT Column-name from table-name WHERE..);
```

IN, NOT IN

Ex:-

```
Select name, rollno, dept   → Column-name
FROM Student                → Table Name
WHERE rollno IN             → expression operator
(Select rollno from Student WHERE dept='CSE');
```

rollno: Column; Student: table name; dept='CSE': condn

name	roll	dept
x	1	CSE
y	2	IT
z	3	CSE
w	4	IT

→

name	roll no	dept
x	1	CSE
z	3	CSE

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Unit-4

PL/SQL:

PL/SQL is a combination of SQL along with the procedural features of programming languages.

- Procedural Language extensions to the Structured Query Language.
- Oracle uses a PL/SQL engine to processes the PL/SQL statements.
- PL/SQL is a block structured language that enables developers to combine the power of SQL with procedural statements.
- PL/SQL includes procedural language elements like conditions and loops.

⇒ functionalites

- PL/SQL is not case sensitive so you are free to use lower case letter or upper case letters. except within string and character literals.
- A line of PL/SQL text contains group of characters known as lexical units
  - Delimeters
  - Literals
  - Identifiers
  - Comments
- /* and */ → Multiline comment.
- -- → Single line Comment
- := assignment
- ; to indicate End of Line

insta - @python_world_in

https://python_world-in.stores.instamojo.com/

# SQL	PL/SQL
• SQL is a single query that is used to perform DML and DCL operations.	• PL/SQL is a block of codes that used to write the entire program blocks/ function etc.
• It is declarative, that defines what need to be done, rather than how things need to be done	• PL/SQL is procedural that defines how the things needs to be done.
• Execute as a single statement	• Execute as a whole block.
• Mainly used to manipulate data.	• Mainly used to create an application.
• Cannot contain PL/SQL code in it	It is an extension of SQL, so it can contain SQL inside it

Advantage of PL/SQL

- Support for SQL → Code reusability → no need to write code again again...
- Performance → oracle engine process multiple SQL statement → looping statement → error handling
- High Security
- Portability
- High productivity
- Support object oriented programming concepts
- PL/SQL is tightly integrated with SQL.
- It Supports the development of web application.
- It offers a variety of programming structure

insta: @python_world_in

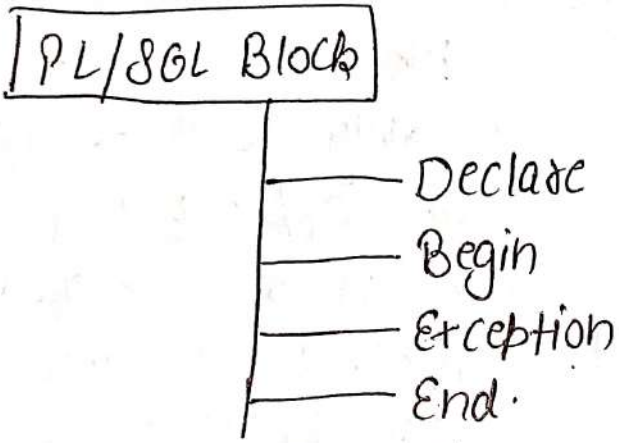
https://python_world_in.stores.instamojo.com/

Disadvantages

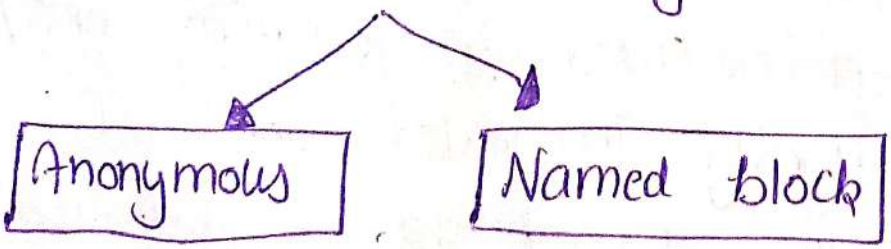
- Requiring specialized knowledge.
- Being limited to Oracle databases
- It uses high memory for storage.
- Lacks debugging functionality

Block/Structure/Genesic in PL/SQL

- The basic unit in PL/SQL is a block.
- All PL/SQL programs are made up of blocks, which are nested within each other.



- A PL/SQL Block is defined by the keywords DECLARE, BEGIN, EXCEPTION, END
- Typically, each block forms a logical action in the program.
- All statements are classified into units that is called blocks.
- PL/SQL block are of two types



insta - @python_world_in

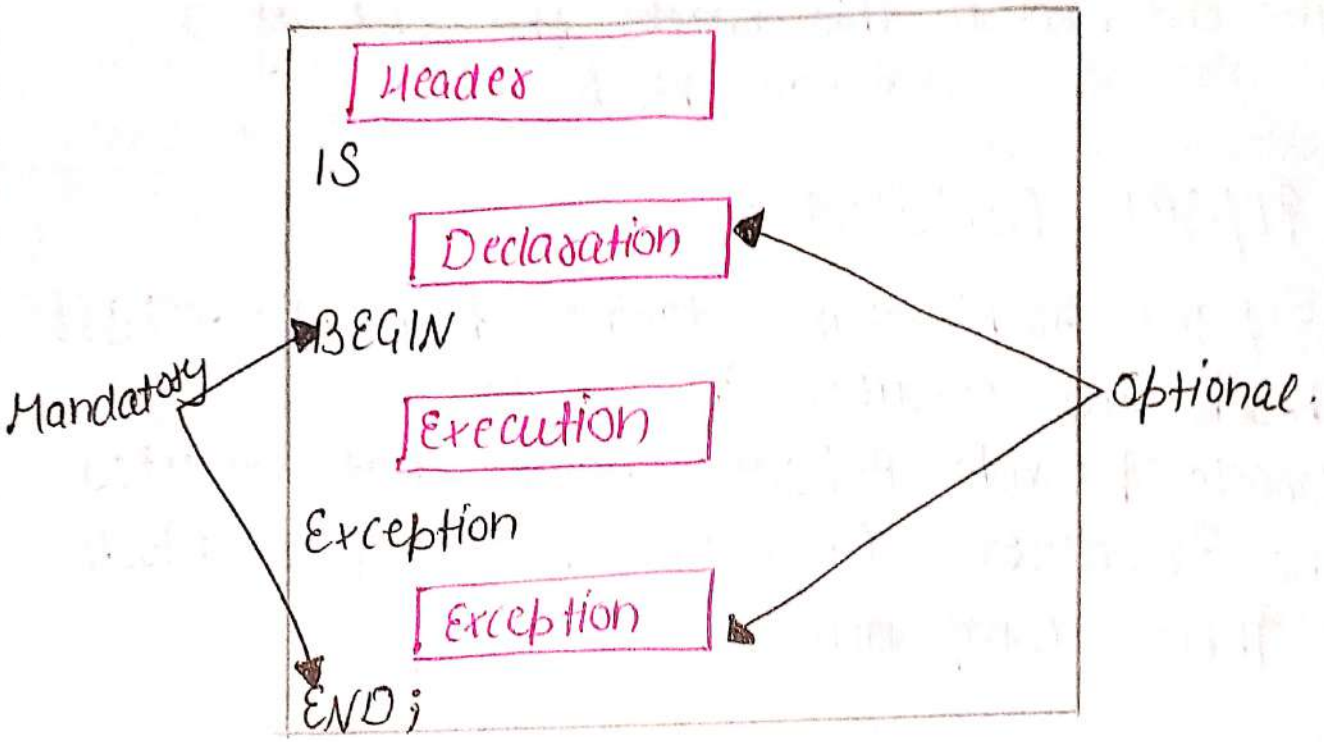
https://pythonworld.in.stores.instamojo.com/

1) Anonymous Block: Block which is not have headers are known as anonym block.

- These blocks do not form the body of a function or triggers or procedure.

2) Named Block: Block which having headers or labels.

- These blocks can either be subprograms (like functions, procedures.



- Declaration Section: This is the first section of the PL/SQL blocks and optional part.

- Code block start with a declaration section, in which memory variables, constants, cursors and other oracle objects can be declared.

- This section starts with keyword 'DELARE'.

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

• Begin Section: This section start with the keyword 'BEGIN'.

• Execution part is the main and mandatory part which actually executes the code that is written inside it.

• Exception Section: This section deal with handling errors that arise during execution

data manipulation statements which make up PL/SQL code block.

• This is an optional section of PL/SQL Blocks.

• The End Section: This marks the end of a PL/SQL block

PL/SQL Architecture

• PL/SQL Architecture defines how the PL/SQL blocks are executed in Oracle.

• Oracle provide PL/SQL engine that executes the Procedural language part of a block.

• Three Components

1) PL/SQL Block

2) PL/SQL Engine.

3) Database Server.

1) PL/SQL Block: → This is the component which has the actual PL/SQL code

• It also contains the SQL instruction that used to interact with the

insta - @python_world_in

https://python-world-in-stores.instamojo.com

database Server.

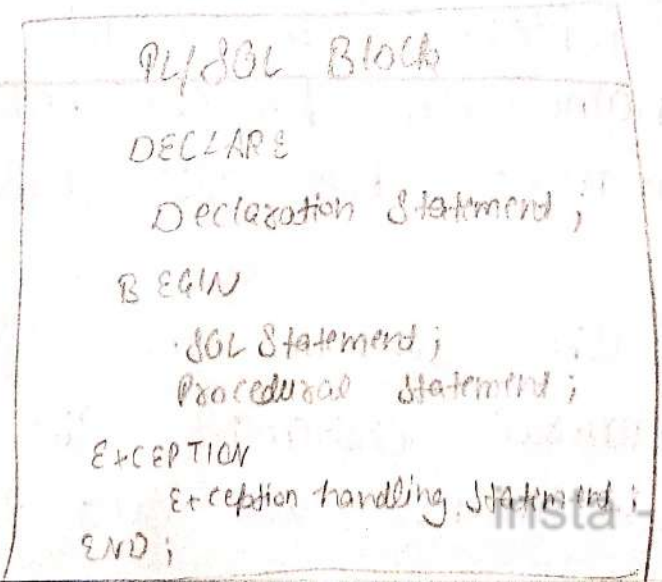
- All the PL/SQL units are treated as PL/SQL Blocks, and this is the starting stage of the architecture which serves as the primary input

2) PL/SQL Engine

- PL/SQL engine is the component where the actual processing of the codes takes place.
- PL/SQL engine separates PL/SQL units and SQL part in the unit
- the PL/SQL engine provides the environment in which the PL/SQL Block is executed

3) Database Server

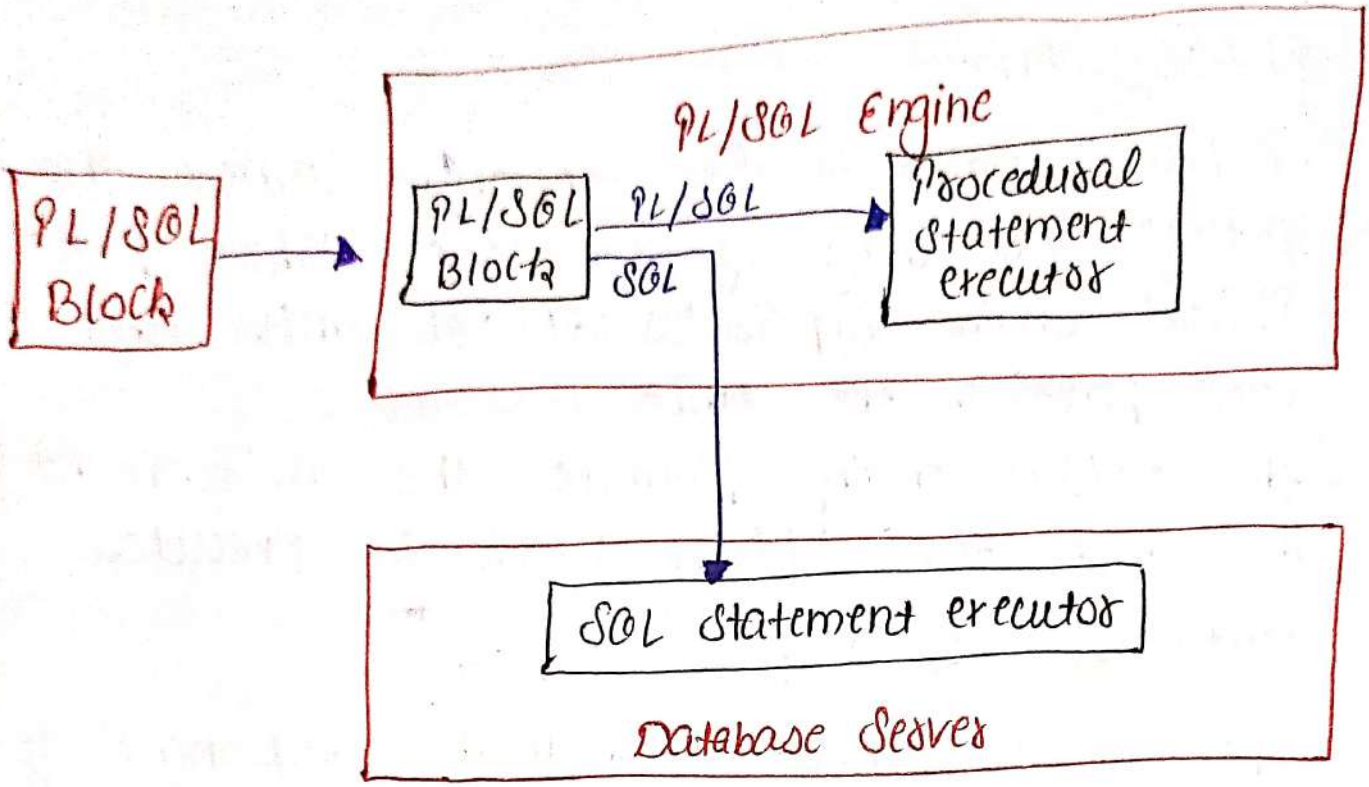
- This is the most important component of PL/SQL unit which stores the data.
- The PL/SQL engine uses the SQL from PL/SQL units to interact with the database server.
- It consists of SQL executor which parses the input SQL statement and execute the same



insta - @python_world_in

https://python-world-in.stores.instamojo.com/

PL/SQL Architecture



PL/SQL Execution Environment

- PL/SQL technology is required the PL/SQL engine accepts any valid PL/SQL block as input
- PL/SQL in the Oracle Engine

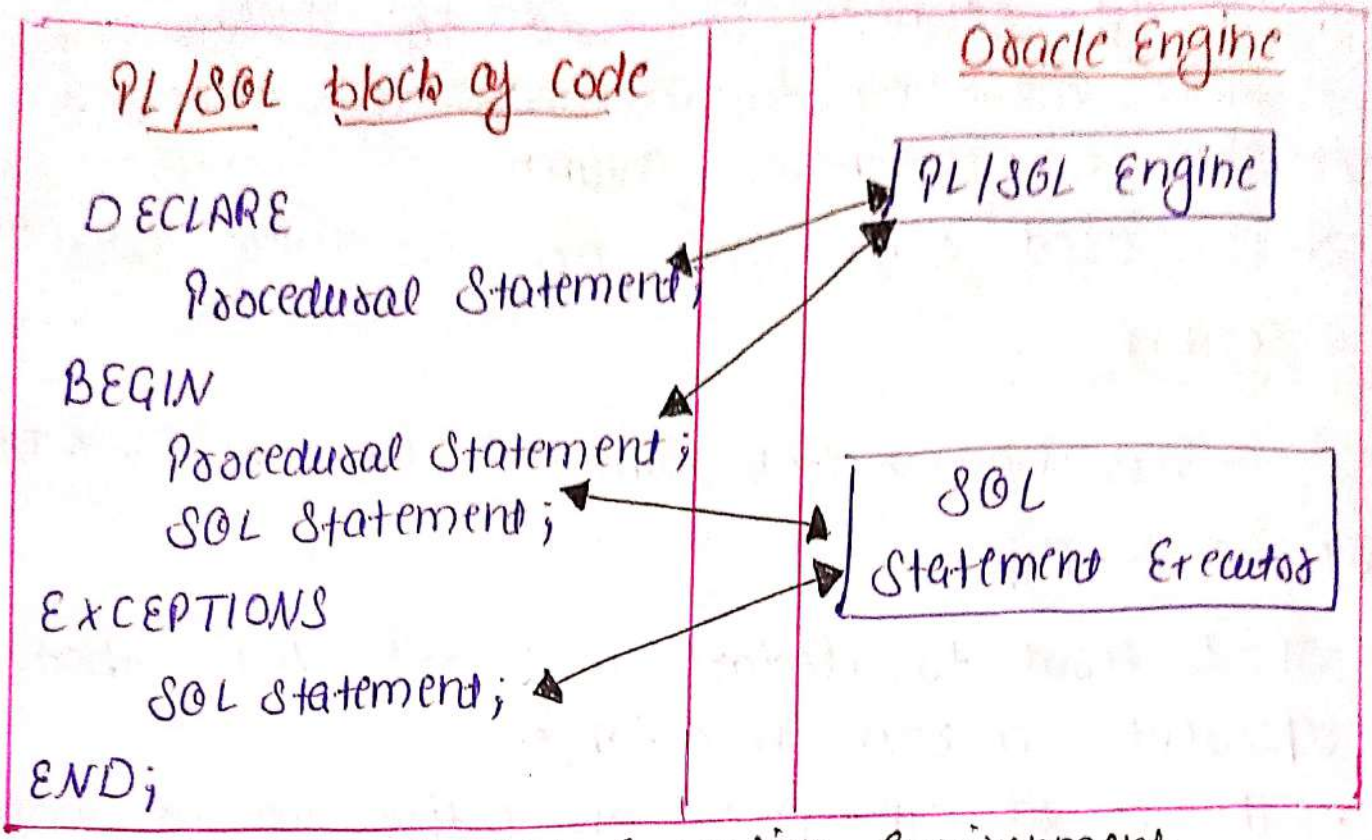
PL/SQL engine resides in the Oracle Engine, the oracle engine can process not only single SQL statement but also entire PL/SQL blocks.

- These block are sent to the PL/SQL engine where procedural Statement are executed. and SQL Statements are sent to the

Insta @python_world_in

SQL executor in the Oracle Engine.

https://python-world-in.stores.instamojo.com/



- The PL/SQL Execution Environment

- It explain how these statement are executed and how convenient it is to bundle SQL code.

Variable Attribute

- When we declare PL/SQL variables to hold column values, it must be of correct datatype and precision, otherwise error will occur during execution. So PL/SQL provides the facility to declare variable without having to specify a particular datatype

1) % Type
2) % RowType

insta - @python_world_in

https://python-world.in-stores.instamojo.com/

1) %Type Attribute

The %Type Attribute is used to declare variable according to the already declared variable or database column.
It is used when you are declaring individu

→ Syntax

<var_name> <tab_name>.<column_name> %TYPE;

2) %ROW Type

It is used to declare a record type that represents a row in a table.
The record can store an entire row or some specific data selected from the table.

→ Syntax:

<var_name> <tab_name>.ROW %TYPE;

→ example: Employee Emp. %ROW TYPE;

Operators in PL/SQL

- Arithmetic [+, -, /, *]
- Relational [True or false]
  [=, !=, <>, <, >]
- logical [AND, OR, NOT]

Comparison [Like, Between, IN, IS NULL]

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Constants

A constant is a value used in a PL/SQL block that remains unchanged throughout the program.

- Syntax:

constant_name CONSTANT datatype := VALUE;

- Constant_name: is the name of constant just like variable name. The constant word is a reserved word and its value does not change
- VALUE: It is a value which is assigned to a constant when it is declared. It cannot be assigned later

Data types

1) Scalar Type :
   - Number
   - Character and String Types
   - Boolean
   - Date, Time.

2) LOB Type :
   - NCLOB
   - CLOB
   - BFILE
   - BLOB

3) Reference Type :
   - REFS
   - REF Cursors

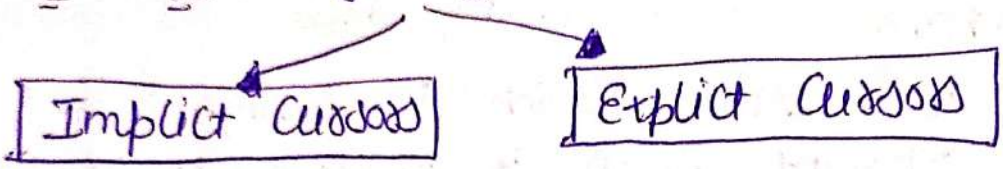
4) Composite Type : Element of Array

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Cursors

- When the SQL statement is processed, Oracle creates a memory area known as context area.
- A cursor is a pointer to this context area.
- It contains all information needed for processing the statement.
- In PL/SQL, the context area is controlled by Cursor. [help in SQL row-wise work.
- Two types of Cursors



1) Implict Cursors: The implict cursors are automatically generated by Oracle while an SQL statement is executed.

- These are created by default to process the statement when DML statement like INSERT, UPDATE, DELETE etc. are executed
- Oracle provides some attributes known as implict cursor's attributes to check the states of DML operation.
  - → %FOUND
  - → %NOT FOUND
  - → %ROW COUNT → returns No of Row effected by DML Statement & returned by SELECT or To Statement
  - → %ISOPEN

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

- %FOUND :- Its return value is TRUE if DML statements like INSERT, DELETE and UPDATE affect at least one row or more rows or a SELECT INTO statement returned one or more rows. Otherwise it returns FALSE.
- %NOTFOUND :- Its return value is TRUE if DML Statement like INSERT, DELETE and UPDATE affect no row. or a SELECT INTO statement return no rows. otherwise it return FALSE.
- %ISOPEN :- It always returns FALSE for implicit cursors, because the SQL cursor is automatically closed after executing its associated SQL Statement
- %ROWCOUNT: It returns the number of rows affected by DML Statement like INSERT or returned by Select into Statement

Implicit	Explicit
• Closed automatically after execution	• Need to be defined explicitly by the user.
• Less efficient	• Need to close after execution.
	• More efficient

insta - @python_world_in

https://python-world.in.stores.instamojo.com/

2) Explicit Cursors

- The explicit cursors are defined by the program to gain more control over the context area.
- These cursors should be defined in the declaration section of the PL/SQL block. It is created on a SELECT Statement which return more than one row.

Syntax

CURSOR cursor-name IS select-statement;;

Steps

1) Declare the cursor:

- It define the cursor with name and the associated SELECT Statement.

Syntax (cursor) → CURSON name IS
SELECT Statement;

2) Open the cursor

- Used to allocate memory for the cursor.

Syntax → OPEN Cursor-name;

) Fetch the cursor

- Used to access one row at a time.

Syntax:→ FETCH cursor-name INTO variable-list;

Close the cursor

- Used to release the allocated memory.
- Syntax:→ Close cursor-name;

insta - @python_world_in

https://python-world-in-stores.instamojo.com/

Syntax:

```
DELARE Variables;
records;
Create a cursor;
BEGIN
OPEN cursor;
FETCH Cursor;
  process the records;
CLOSE cursor;
END;
```

Example:

ID	Name	Age	Address
1	R	23	Delhi
2	S	22	Kanpur
3	M	24	Ghaziabad
4	C	25	Noida

- Create Procedure

```
DECLARE
  c_id customers.id %type;
  c_name customer.name %type;
  c_addr customers.address %type;
CURSOR c_customer is
  SELECT id, name, address FROM customers;
BEGIN
  OPEN c_customers;
  Loop
  FETCH c_customers into c_id, c_name, c_addr;
  EXIT WHEN c_customer %notfound;
  dbms_output.put_line (c_id || ' ' || c_name || ' ' ||
                        c_addr);
```

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

```
END LOOP;
    CLOSE C_customers;
END;
/
```

Output:
```
1  R  Delhi
2  S  Kanpur
3  M  Ghaziabad
4  C  Noida.
```
PL/SQL procedure successfully completed.

Implicit Cursor Example

ID	Name	Age	Address	Salary
1	A	28	∞	20000
2	B	29	∞	22000
3	C	30	∞	24000

- To update the table and increase salary of each customer by 5000.

```
DECLARE
    total_rows number(2);
BEGIN
    UPDATE customers
    SET salary = salary + 5000;
    IF SQL%notfound THEN
      dbms_output.put_line('no customer updated');
    ELSEIF SQL%FOUND THEN
      total_rows := SQL%rowcount;
    dbms_output.put_line(total rows || ' customer updated');
  END IF;
END;
```

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Output = 3 Customers Updated.

```
SELECT * FROM CUSTOMERS;
```

ID	NAME	AGE	ADDRESS	SALARY
1	A	29	∞	25000
2	B	30	∞	27000
3	C	30	∞	29000

Triggers in PL/SQL

- Trigger is invoked by Oracle engine automatically whenever a specified event occurs.
- Trigger are stored programs, which are automatically executed or fired when some event occurs.
- Trigger are written to be executed in respon to any of the following events.
  - a database manipulation DML Statemen (INSERT, UPDATE, DELETE)
  - a Database definition DDL Statement (CREATE, ALTER, DROP)
  - a database operation. (LOGON, LOGOFF, STARTUP and STARTDOWN)
- Triggers could be defined on the table, view, schema or database with which the event is associated.

insta - @python_world_in

https://python-world-in.stores.instamojo.com/

Advantages

- Triggers generates some derived column values automatically
- Auditing
- Enforces referential integrity
- Event logging and storing information on table access

→ Categories of Trigger level

Row level
- It get executed for each record that got updated by a DML statement

Statement level
- It get executed only once by the event statement.

→ Categories of trigger timing

Before | After | INSTEAD of Trigger

→ Categories of trigger event

DML | DDL | Database.

→ :NEW: It holds a new value for the column of the base table/view during the trigger execution.

→ :OLD: It hold old value of the column of the base table/view during the trigger execution.

Insta @python_world_in

https://python_world-freebies.instamojo.com/

→ Creating a trigger

```
CREATE [OR REPLACE] TRIGGER trigger_name
[BEFORE | AFTER | INSTEAD OF]   → Trigger Timing
[INSERT | UPDATE | DELETE ..]   → Event
[OF Col_name]
ON table_name.
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]   ← Row level.
WHER [Condition)
DECLARE
        Declaration-statements
BEGIN
        Executable - Statements
EXCEPTION
        Exception - handling - Statements
END;.
```

- Create Trigger_name: Creates an existing trigger with Trigger_name
- Before / After: This specifies when the trigger would be executed.
- Insert / Update / Delete: This specifies the DML oper
- Of Col-name: This specifies the column name tha would be updated.
- On table-name: This specifies the name of table associated with the trigger.
- Referencing old as o NEW as n: This allow you to refer new and old values for various DML statement.

insta @python_world_in

https://python-world-in-stores.instamojo.com/

Ex:

Id	Name	Age	Address	Salary
1	A	23	abc	20000
2	B	22	bcd	22000
3	C	24	cdf	24000
4	D	25	Gdll	26000

- CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON Customer

```
FOR each row
WHEN (NEW.id > 0)
DECLARE
    sal_diff number;
BEGIN
    sal_diff := :NEW.salary - :OLD salary;
dbms_output.put_line ('old salary:' || :OLD.salary);
dbms_output.put_line ('NEW salary:' || :NEW.salary);
dbms_output.put_line ('salary diff :' || sal_diff);
END;
/
```

Output:

```
old salary : 2000
New salary : 25000
Salary difference : 5000
```

insta - @python_world_in